Computational Linguistics and Chinese Language Processing
Vol. 17, No. 4, December 2012, pp. 1-14

Detecting and Correcting Syntactic Errors in Machine Translation Using Feature-Based Lexicalized Tree Adjoining Grammars

Wei-Yun Ma*, and Kathleen McKeown*

Abstract

Statistical machine translation has made tremendous progress over the past ten years. The output of even the best systems, however, is often ungrammatical because of the lack of sufficient linguistic knowledge. Even when systems incorporate syntax in the translation process, syntactic errors still result. To address this issue, we present a novel approach for detecting and correcting ungrammatical translations. In order to simultaneously detect multiple errors and their corresponding words in a formal framework, we use feature-based lexicalized tree adjoining grammars, where each lexical item is associated with a syntactic elementary tree, in which each node is associated with a set of feature-value pairs to define the lexical item's syntactic usage. Our syntactic error detection works by checking the feature values of all lexical items within a sentence using a unification framework. In order to simultaneously detect multiple error types and track their corresponding words, we propose a new unification method which allows the unification procedure to continue when unification fails and also to propagate the failure information to relevant words. Once error types and their corresponding words are detected, one is able to correct errors based on a unified consideration of all related words under the same error types. In this paper, we present some simple mechanism to handle part of the detected situations. We use our approach to detect and correct translations of six single statistical machine translation systems. The results show that most of the corrected translations are improved.

Keywords: Machine Translation, Syntactic Error, Post Editing, Tree Adjoining Grammar, Unification.

* Department of Computer Science, Columbia University, New York, USA
E-mail: {ma, kathy}@cs.columbia.edu

1. Introduction

Statistical machine translation has made tremendous progress over the past ten years. The output of even the best systems, however, is often ungrammatical because of the lack of sufficient linguistic knowledge. Even when systems incorporate syntax in the translation process, syntactic errors still result. We have developed a novel, post-editing approach which features: 1) the use of XTAG grammar, a rule-based grammar developed by linguists, 2) the ability to simultaneously detect multiple ungrammatical types and their corresponding words by using unification of feature structures, and 3) the ability to simultaneously correct multiple ungrammatical types based on the detection information. To date, we have developed the infrastructure for this approach and demonstrated its utility for agreement errors.

As illustrative examples, consider the following three ungrammatical English sentences:

1. Many young student play basketball.
2. John play basketball and Tom also play basketball.
3. John thinks to play basketball.

In 1 and 2 above, number agreement errors between the subjects and verbs (and quantifier) cause the sentences to be ungrammatical, while in 3, the infinitive following the main verb makes it ungrammatical. One could argue that an existing grammar checker could do the error detection for us, but if we use Microsoft Word 2010 (MS Word)'s grammar checker (Heidorn, 2000) to check the three sentences, the entire first sentence will be underlined with green wavy lines without any indication of what should be corrected, while no errors are detected in 2 and 3.

The grammar we use is based on a feature-based lexicalized tree adjoining grammars (FB-LTAG) English grammar, named XTAG grammar (XTAG group, 2001). In FB-LTAG, each lexical item is associated with a syntactic elementary tree, in which each node is associated with a set of feature-value pairs, called Attribute Value Matrices (AVMs). AVMs define the lexical item's syntactic usage. Our syntactic error detection works by checking the AVM values of all lexical items within a sentence using a unification framework. Thus, we use the feature structures in the AVMs to detect the error type and corresponding words. In order to simultaneously detect multiple error types and track their corresponding words, we propose a new unification method which allows the unification procedure to continue when unification fails and also to propagate the failure information to relevant words. We call the modified unification a *fail propagation unification.*

2. Related Work

Grammar checking is mostly used in word processors as a writing aid. Three methods are widely used for grammar checking given a sentence: statistic-based checking, rule-based checking and syntax-based checking. In statistic-based checking, POS tag sequences (Atwell & Elliot, 1987) or an N-gram language model (Alam *et al*., 2006; Wu *et al*., 2006) is trained from a training corpus and uncommon sequences in the training corpus are considered incorrect. Huang *et al*. (2010) extracted erroneous and correct patterns of consecutive words from the data of an online-editing diary website. In rule-based checking, a set of hand crafted rules out of words, POS tags and chucks (Naber, 2003) or parsing results (Heidorn, 2000) are designed to detect errors. In syntax-based checking, Jensen *et al*. (1993) utilize a parsing procedure to detect errors: each sentence must be syntactically parsed; a sentence is considered incorrect if parsing does not succeed.

Focusing on machine translation's grammar checking, Stymne and Ahrenberg (2010) utilized an existing rule-based Swedish grammar checker, as a post-processing tool for their English-Swedish translation system. They tried to fix the ungrammatical translation parts by applying the grammar checker's correction suggestions. In contrast of their using an existing grammar checker, we developed our own novel grammar checker for translated English in order to better controlling the quality of error detection, error types, and the directions of error correction in translation context.

Our approach is a mix of rule-based checking and syntax-based checking: The XTAG English grammar is designed by linguists while the detecting procedure is based on syntactic operations which dynamically reference the grammar. The work could be regarded as an extension of (Ma & McKeown, 2011), in which grammatical error detection based on XTAG English grammar is carried out to filter out ungrammatical combined translations in their framework of system combination for machine translation. In contrast of (Ma & McKeown, 2011), our approach is not only capable to detect grammatical errors, but also has the capability of identifying error types and errors' causes, and correcting certain cases of errors.

3. Background

We briefly introduce the FB-LTAG formalism and XTAG grammar in this section.

3.1 Feature-Based Lexicalized Tree Adjoining Grammars

FB-LTAG is based on tree adjoining grammar (TAG) proposed in (Joshi *et al*., 1975). The TAG formalism is a formal tree rewriting system, which consists of a set of elementary trees, corresponding to minimal linguistic structures that localize the dependencies, such as specifying the predicate-argument structure of a lexeme. Elementary trees are divided into

initial and auxiliary trees. Initial trees are those for which all non-terminal nodes on the frontier are substitutable, marked with "↓". Auxiliary trees are defined as initial trees, except that exactly one frontier, nonterminal node must be a foot node, marked with "*", with the same label with the root node. Two operations - substitution and adjunction are provided in TAG to adjoin elementary trees.

FB-LTAG has two important characteristics: First, it is a lexicalized TAG (Schabes, 1988). Thus each elementary tree is associated with at least one lexical item. Second, it is a feature-based lexicalized TAG (Vijay-Shanker & Joshi, 1988). Each node in an elementary tree is constrained by two sets of feature-value pairs (two AVMs). One AVM (top AVM) defines the relation of the node to its super-tree, and the other AVM (bottom AVM) defines the relation of the node to its descendants. We use Fig1 and Fig2[1] to illustrate the substitution and adjunction operations with the unification framework respectively.

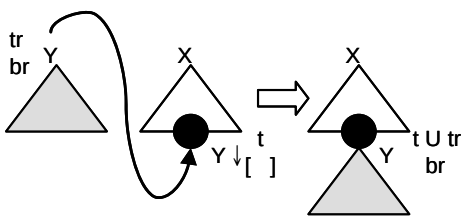

Figure 1. Substitution of FB-LTAG

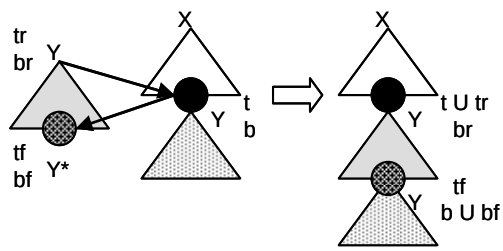

Figure 2. Adjunction of FB-LTAG

In Fig 1, we can see that the feature structure of a new node created by substitution inherits the union of the features of the original nodes. The top feature of the new node is the union of the top features of the two original nodes, while the bottom feature of the new node is simply the bottom feature of the top node of the substituting tree. In Fig 2, we can see that the node being adjoined into splits, and its top feature unifies with the top feature of the root adjoining node, while its bottom feature unifies with the bottom feature of the foot adjoining node.

3.2 XTAG English Grammar

XTAG English grammar (XTAG group, 2001) is designed using the FB-LTAG formalism, released[2] by UPENN in 2001. The range of syntactic phenomena that can be handled is large. It defines 57 major elementary trees (tree families) and 50 feature types, such as agreement, case, mode (mood), tense, passive, etc, for its 20027 lexical entries. Each lexical entry is

[1] The two figures and their descriptions are based on the XTAG technical report (XTAG group, 2001)

[2] http://www.cis.upenn.edu/~xtag/gramrelease.html

associated with at least one elementary tree, and each elementary tree is associated with at least one AVM. For example, Fig 3 shows the simplified elementary tree of "saw". "<number>" indicates the same feature value. For example, the feature – "arg_3rdsing" in bottom AVM of root S should have the same feature value of "arg_3rdsing" in top AVM of VP. In our implementation, it is coded using the same object in an object-oriented programming language. Since the feature value of mode in top AVM of "S ↓ " is "base", we know that "saw" can only be followed by a sentence with a base verb. For example, "He saw me do that" shown in Fig 4(a) is a grammatical sentence while "He saw me to do that" shown in Fig 4(b) is an ungrammatical sentence because "saw" is not allowed to be followed by an infinitive sentence.

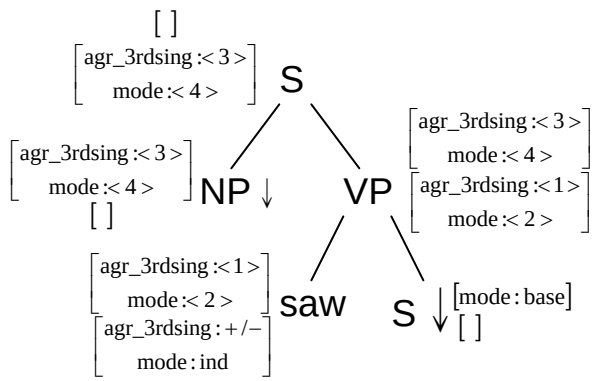

Figure 3. Elementary tree for "saw"

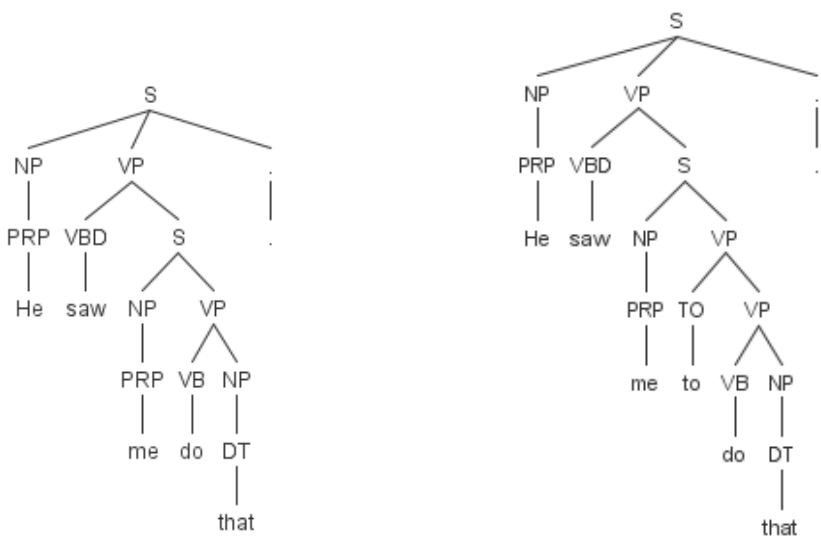

Figure 4(a). Grammatical sentence of "saw" (b) Ungrammatical sentence of "saw"

But if we look at the simplified elementary tree of "asked" shown in Fig 5, we can find that "asked" can only be followed by a sentence with an infinitive sentence (inf). For example, "He asked me to do that" shown in Fig 6(a) is a grammatical sentence while "He asked me do that" shown in Fig 6(b) is an ungrammatical sentence because "asked" is not allowed to be followed by a sentence with a base verb.

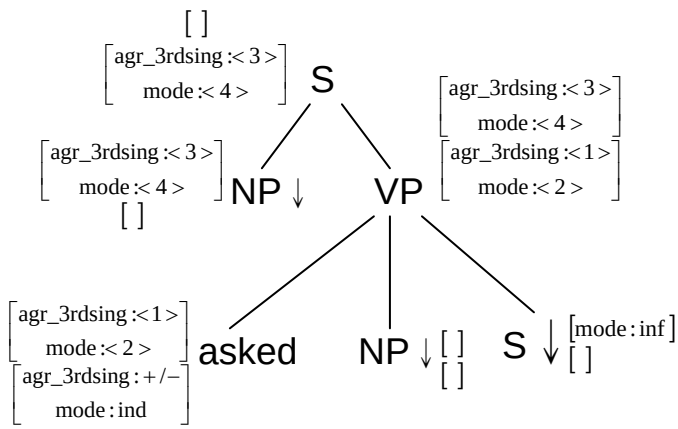

Figure 5. Elementary tree for "asked"

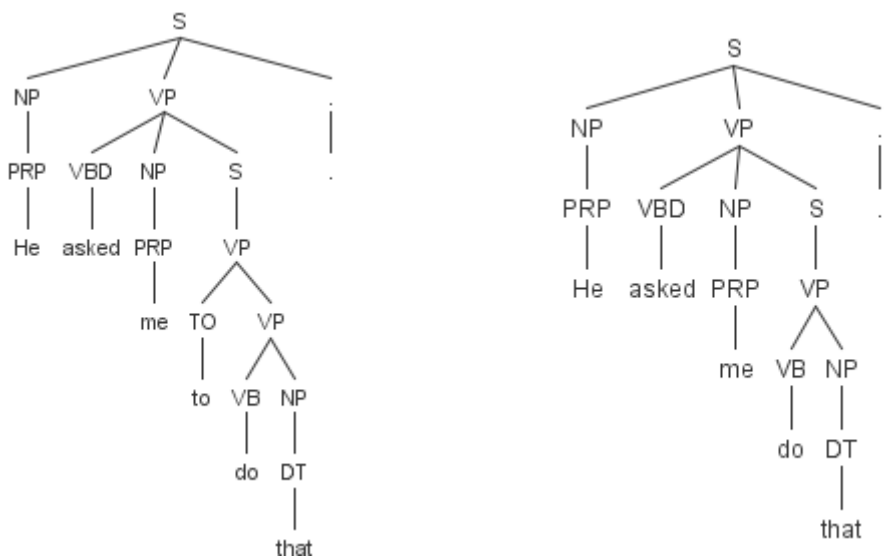

Figure 6(a). Grammatical sentence of "asked"(b) Ungrammatical sentence of "asked"

4. Syntactic Error Detection

Our procedure for syntactic error detection includes 1. decomposing each sentence hypothesis parse tree into elementary trees, 2. associating each elementary tree with AVMs through look-up in the XTAG grammar, and 3. reconstructing the original parse tree out of the elementary trees using substitution and adjunction operations along with AVM unifications.

When unification of the AVMs fails, a grammatical error has been detected and its error type is also identified by the corresponding feature in the AVM. In order to simultaneously detect multiple error types and their corresponding words, we adjust the traditional unification definition to allow the unification procedure to continue after an AVM failure occurs and also propagate the failure information to relevant words. We call the modified unification *fail propagation unification*.

Each step is illustrated in this section.

4.1 Decomposing to Elementary trees

Given a translation sentence, we first get its syntactic parse using the Stanford parser (Klein & Manning, 2003) and then decompose the parse to multiple elementary trees by using an elementary tree extractor, a modification of (Chen & Vijay-Shanker, 2000). After that, each lexical item in the sentence will be assigned one elementary tree. Taking the sentence – "Many young student play basketball" as an example, its parse and extracted elementary trees are shown in Fig 7 and Fig 8, respectively. In Fig 8, the arrows represent relations among the elementary trees and the relations are either substitution or adjunction. In this example, the two upper arrows are substitutions and the two bottom arrows are adjunctions.

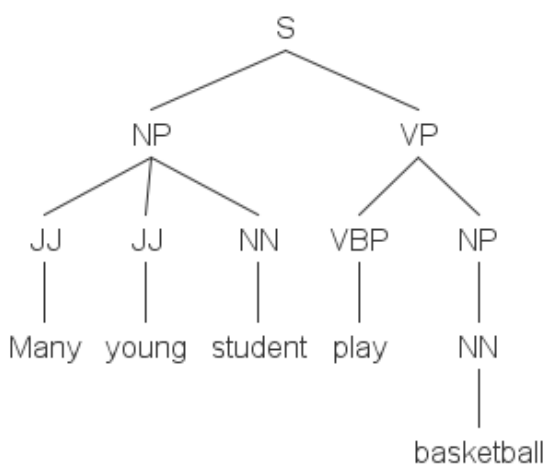

Figure 7. Parse of "Many young student play basketball"

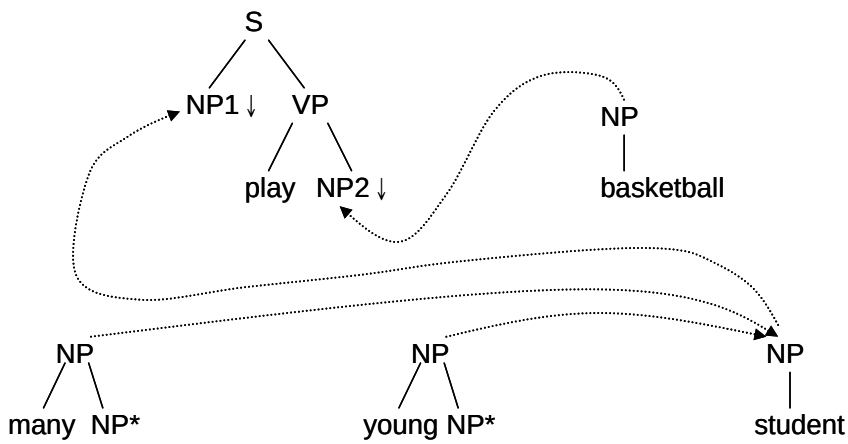

Figure 8. The elementary trees of 'Many young student play basketball" and their relations

4.2 Associating AVMs to Elementary trees

Each elementary tree is associated with AVMs through look-up in the XTAG English grammar. Using the same example of the sentence – "Many young student play basketball", its elementary trees, relations and one set of AVMs (simplified version) are shown in Fig 9. To keep tracing what word(s) that a feature value relates to for the next step of reconstruction, we design a new data structure of word set, named "word trace". It is represented by "{…}" and attached with each feature value except the value of "null", such as "agr_num:pl{play}" in Fig 9.

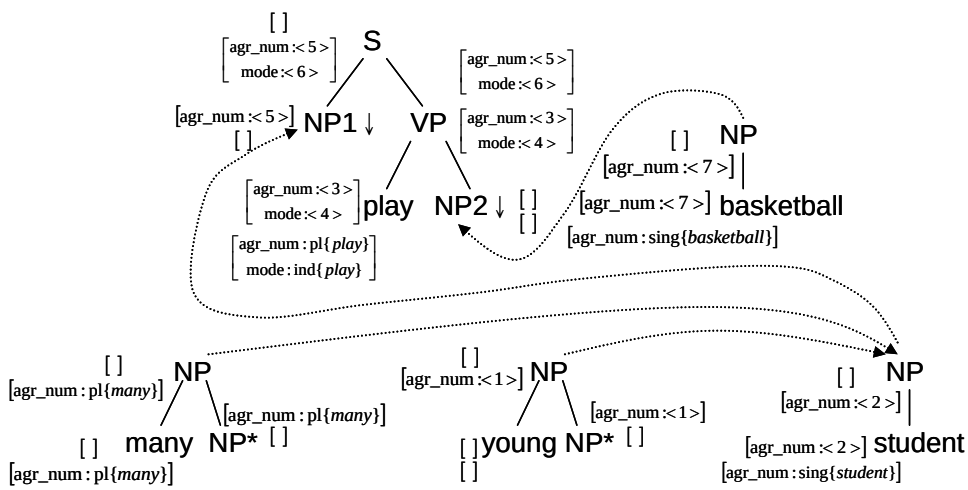

Figure 9. The elementary trees of 'Many young student play basketball", their relations and AVMs (simplified version).

When we loop up the XTAG English Grammar, sometimes one elementary tree could have multiple possible AVM associations. For example, for the verb “are”, one of its elementary trees is associated with three different AVMs, one for 2nd person singular, one for 2nd person plural, and one for 3rd person plural. Unless we can reference the context for “are” (e.g., its subject), we are not sure which AVM should be used in the reconstruction. So we postpone this decision until later in the reconstruction process. At this point, we associate each elementary tree with its all possible AVMs defined in the XTAG English Grammar.

4.3 Reconstruction Framework

Once the elementary trees are associated with AVMs, they will be used to reconstruct the original parse tree through substitution and adjunction operations which are indicated during the process of decomposing a parse tree to elementary trees. The reconstruction process is able to decide if there is any conflict with the AVMs values. When a conflict occurs, it will cause an AVM unification failure, referring to a certain grammatical error.

We already illustrated how substitution and adjunctions along with AVM unifications work in section 3.1; one implementation complement is, once the original parse is constructed, it is necessary to unify every node’s top and bottom AVMs in the constructed tree. This is because, in XTAG grammar, most AVM values are assigned in the anchor nodes of elementary trees and were not unified with others yet. This end step will assure that all related AVMs are unified.

As we stated in Section 4.2, sometimes we are not sure which AVM association for one elementary tree should be used in the reconstruction. So our strategy is to carry out reconstruction process for all sets out of every elementary tree’s each possible AVM association. We choose the set that causes the minimal grammatical errors as the detection result.

4.4 Fail Propagation Unification

Our system detects grammatical errors by identifying unification fails. However, traditional unification does not define how to proceed after fails occur, and also lacks an appropriate structure to record error traces. So we extend it as follows:

[f=x] {t_1}	U	[f=x] {t_2}	=>	[f=x] {t_1} union {t_2}	(1)
[f=x] {t_1}	U	[f=null]	=>	[f=x] {t_1}	(2)
[f=null]	U	[f=null]	=>	[f=null]	(3)
[f=x] {t_1}	U	[f=y] {t_2}	=>	[f=fail] {t_1} union {t_2}	(4)
[f=fail] {t_1}	U	[f=null]	=>	[f=fail] {t_1}	(5)
[f=fail] {t_1}	U	[f=y] {t_2}	=>	[f=fail] {t_1} union {t_2}	(6)
[f=fail] {t_1}	U	[f=fail] {t_2}	=>	[f=fail] {t_1} union {t_2}	(7)

Where f is a feature type, such as “arg_num”; x and y are two different feature values; U represents the “unify” operation; t1 and t2 are word traces introduced in section 4.2. “fail” is also defined as a kind of value.

(1)~(4) are actually traditional unification definitions except that the word trace union operations and the characteristic of fail have been added. When a unification failure occurs in (4), the unification procedure does not halt but only assigns f a value of fail and proceeds. (5)~(7) propagate the fail value to the related words’ AVMs. We use the following two unifications occurring in order in Fig 9’s adjoining operations to illustrate the procedure of fail propagation unification:

[arg_num=pl]{many} U [arg_num=sing]{student}
=> [arg_num =fail]{many,student}

[arg_num=fail]{many, student} U [arg_num=pl]{play}
=> [arg_num =fail]{many,student,play}

By the feature value of “fail” and the word trace, we identify that there is an agr_num error related to three words – “many”, “student” and “play”.

All AVMs in Fig 9 after unifications along with reconstruction operations are shown in Fig 10.

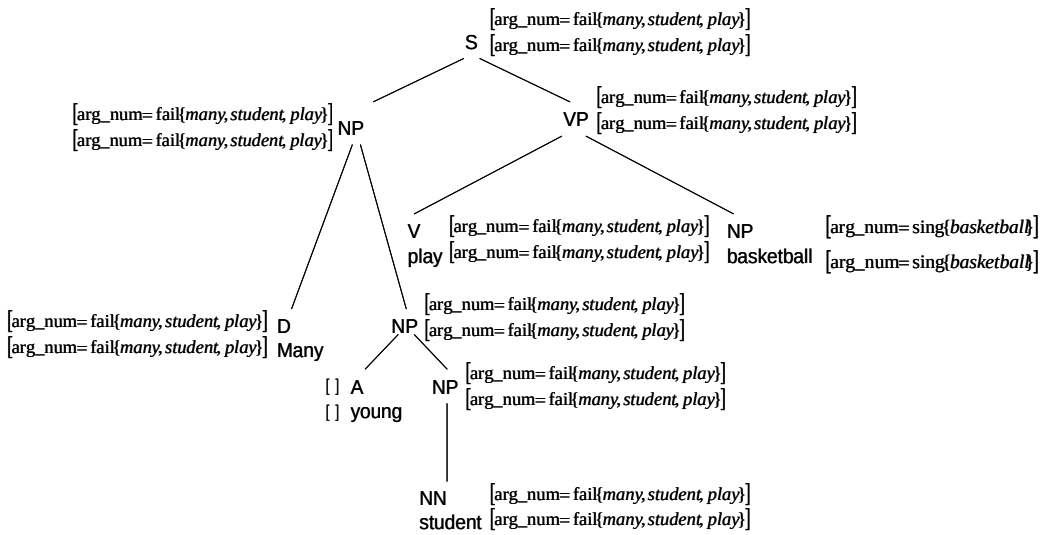

Figure 10. Reconstructed parse of the sentence- "Many young student play basketball" after unifications with fail propagation

5. Syntactic Error Correction

Once error types and their corresponding words are detected, one is able to correct errors based on an unified consideration of all related words under the same error types.

Given a set of related ungrammatical words, there are two tasks for the correction process: which words should be corrected and how to correct them? To date, we have developed the following simple mechanism to handle the agreement problem: First, the words whose feature value is in the minority will be selected to be corrected. We call this feature-value voting. Take the above example: "student" should be corrected since its agr_num is "sing" and the other two words' agr_num is "plural". When facing cases of equal votes, we tend to correct nouns if there are nouns.

Once the corrected words are selected, we replace them with their variations but with the same elementary tree type, such as replacing the above "student" with "students."

6. Experiment

Among the 57 major elementary trees and 50 feature types that XTAG defines, we have implemented 26 major elementary trees and 4 feature types – agr_pers, arg_num, arg_3rdsing and several cases of mode/mood at this point (The first three belong to agreement features.) We apply our syntactic error detection and correction on 422 translation sentences of six Chinese-English machine translation systems A~F from the DARPA Global Autonomous Language Exploitation (GALE) 2008 evaluation. Every source sentence is provided along

with four target references. The six systems are described in Table 1, and the results of syntactic error detection for agreement and mode errors and correction for agreement errors are shown in Table 2.

Table 1. Six MT systems

	System name	Approach
A	NRC	phrase-based SMT
B	RWTH-PBT	phrase-based SMT
C	RWTH-PBT-AML	phrase-based SMT + source reordering
D	RWTH-PBT-JX	phrase-based SMT + Chinese word segmentation
E	RWTH-PBT-SH	phrase-based SMT + source reordering + rescoring
F	SRI-HPBT	hierarchical phrase-based SMT

Table 2. The results of syntactic error detection and correction

	Detected sentences (arg error + mode error)	Corrected sentences (arg error)	Bleu for all sentences (before)	Bleu for all sentences (after)	Bleu for corrected sentences (before)	Bleu for corrected sentences (after)
A	23	9	32.99	32.99	26.75	27.80
B	23	14	27.95	27.97	22.08	23.03
C	18	7	34.40	34.41	32.13	32.67
D	25	14	32.96	32.99	31.49	32.17
E	30	11	34.64	34.68	29.31	30.61
F	18	8	34.13	34.14	29.15	28.83

From Table 2, even the overall Bleu score for all sentences is not significantly improved, but if we take a close look at those corrected sentences for agreement errors and calculate their Bleu scores, we can see the corrected translations are improved for every system except for one (F), which shows the effectiveness and potential of our approach.

7. Conclusion

This paper presents a new FB-LTAG-based syntactic error detection and correction mechanism along with a novel AVM unification method to simultaneously detect multiple ungrammatical types and their corresponding words for machine translation. The mechanism can also be applied to other languages if the grammar is well defined in the FB-LTAG structure of certain languages.

While the basic design philosophy and algorithm are fully described in this paper, we are continuing to implement more elementary trees and feature types defined in the XTAG grammar, and we are extending our correction mechanism as our future work.

Acknowledgments

We would like to thank the anonymous reviewers for their helpful comments. This work is supported by the National Science Foundation via Grant No. 0910778 entitled "Richer Representations for Machine Translation". All views expressed in this paper are those of the authors and do not necessarily represent the view of the National Science Foundation.

Reference

Alam, M. J., UzZaman, N. & Khan, M. (2006). N-gram based Statistical Grammar Checker for Bangla and English. In *Proceedings of ninth International Conference on Computer and Information Technology (ICCIT 2006)*, Dhaka, Bangladesh.

Atwell, E. S. & Elliot, S. (1987). Dealing with Ill-formed English Text. In: R. Garside, G. Leech and G. Sampson (Eds.) *The Computational Analysis of English: A Corpus-based Approach*. London: Longman.

Chen, J. & Vijay-Shanker, K. (2000). Automated extraction of TAGs from the Penn treebank. In *Proceedings of the Sixth International Workshop on Parsing Technologies*.

Heidorn, G. E. (2000). Intelligent writing assistance. In R. Dale, H. Moisl and H. Somers (eds.), *A Handbook of Natural Language Processing: Techniques and Applications for the Processing of Language as Text*. Marcel Dekker, New York. 181-207.

Huang, A., Kuo, T. T., Lai, Y. C. & Lin, S. D. (2010). Identifying Correction Rules for Auto Editing. In *Proceedings of the 22nd Conference on Computational Linguistics and Speech Processing (ROCLING)*, 251-265.

Jensen, K., Heidorn, G. E. & Richardson, S. D. (Eds.) (1993). *Natural Language Processing: The PLNLP Approach*, Kluwer Academic Publishers.

Joshi, A. K., Levy, L. S. & Takahashi M. (1975). Tree Adjunct Grammars. *Journal of Computer and System Science*, 10, 136-163.

Klein, D. & Manning, C. D. (2003). Accurate Unlexicalized Parsing. In *Proceedings of the 41st Meeting of the Association for Computational Linguistics*, 423-430.

Ma, W. Y. & McKeown, K. (2011). System Combination for Machine Translation Based on Text-to-Text Generation. In *Proceedings of Machine Translation Summit XIII*. Xiamen, China.

Naber, D. (2003). A Rule-Based Style and Grammar Checker. *Diploma Thesis*. University of Bielefeld, Germany.

Schabes, Y., Abeille, A. & Joshi, A. K. (1988). Parsing strategies with 'lexicalized' grammars: Application to tree adjoining grammars. In *Proceeding of 12th International Conference on Computational Linguistics (COLING'88)*, Budapest, Hungary.

Stymne, S. & Ahrenberg, L. (2010). Using a Grammar Checker for Evaluation and Postprocessing of Statistical Machine Translation. In *Proceedings of International Conference on Language Resources and Evaluation (LREC).*

Vijay-Shanker, K. & Joshi, A. K. (1988). Feature structure based tree adjoining grammar, In *Proceeding of 12th International Conference on Computational Linguistics (COLING'88)*, 714-719.

Wu, S. H., Su, C. Y., Jiang, T. J. & Hsu, W. L. (2006). An Evaluation of Adopting Language Model as the Checker of Preposition Usage. In *Proceedings of the Conference on Computational Linguistics and Speech Processing (ROCLING).*

XTAG Group. (2001). A Lexicalized Tree Adjoining Grammar for English. *Technical Report IRCS 01-03*, University of Pennsylvania.